Actividad 1:

1-. Como vimos en clases, este es el coeficiente que nos indica cuánto el algoritmo explorará frente a explotará el conocimiento que ya posee. Es un parámetro que nosotros podemos modificar según la situación, y que influye directamente en cómo el agente interactuará con el ambiente.

2-. Si el “exploration rate” es muy bajo, el agente podría quedarse sin explorar nuevas opciones de camino que resultaban ser más óptimas o de mejor recompensa, quedándose la mayor parte del tiempo mapeando caminos ya conocidos. En cambio, el “exploration decay rate” es la tasa a la cual se va ajustando (disminuyendo) este nivel de exploración a lo largo del tiempo, cambiando la tasa actual por un factor de decaimiento (que suele ir entre 0 y 1) para tener un nuevo valor de exploration rate. Como vimos en clases, este se va reduciendo a medida que el agente mapea las opciones, pues cada vez es capaz de aprender menos, y confía más en su conocimiento. Entonces, si este es muy alto, el exploration rate disminuirá lentamente, lo que podría llevar al agente a explorar más tiempo del necesario, teniendo ya opciones más óptimas.

3-. Por otro lado, si es muy bajo, el agente explorará menos tiempo del necesario, quedándose con opciones menos óptimas probablemente.

Se me ocurre que el juego comenzará probando ejecutar acciones en un inicio, e irá aprendiendo que si le llega un estado con cierta distancia en x, cierta distancia en y y cierta dirección en y, le puede convenir saltar o no, así irá mapeando distintos estados y probando distintas acciones hasta aprender cómo reaccionar de forma óptima a cada estado.

Actividad 3:

1-. Esta nos indica cuánto vale la recompensa más a futuro. Como vimos en clases, si esta es alta, entonces la recompensa a futuro es importante, si es baja, lo contrario. Esto permite que el algoritmo trate de buscar la mejor solución lo antes posible si es que su valor es bajo, por en contrario, si en cambio nos importa que a lo largo del juego se mantenga siendo óptimo (como podría ser en el caso del ajedrez), necesitamos uno alto, de tal forma que siga siendo importante la recompensa de lo que pasa muy a futuro.

2-. Muestro a continuación los resultados cada 10000 iteraciones con distintas tasa de descuento:

- tasa de descuento: 0 y LR: 0.3. Opciones por defecto.

Game 1000 Mean Score 2.01 Record: 4 EXP_RATE: 0.8608649248865096 STEPS: 1868.1

Game 10000 Mean Score 28.8 Record: 20 EXP_RATE: 0.22331899945635536 STEPS: 5266.78

Game 20000 Mean Score 183.57 Record: 131 EXP_RATE: 0.049984578080854006 STEPS: 23113.63

Game 30000 Mean Score 577.65 Record: 179 EXP_RATE: 0.011308440880102967 STEPS: 62783.41

Game 40000 Mean Score 985.71 Record: 185 EXP_RATE: 0.0026786281925767403 STEPS: 99591.76

Vemos que cerca de 30000 en este caso ya no tiene gran mejoría en el record, así que las comparaciones las haremos respecto a este número de iteraciones.

- tasa de descuento: 0.5

Game 1000 Mean Score 1.79 Record: 3 EXP_RATE: 0.8608649248865096 STEPS: 1836.15

Game 10000 Mean Score 98.18 Record: 23 EXP_RATE: 0.22331899945635536 STEPS: 29133.0

Game 20000 Mean Score 833.25 Record: 114 EXP_RATE: 0.049984578080854006 STEPS: 117017.85

Game 30000 Mean Score 3385.84 Record: 178 EXP_RATE: 0.011308440880102967 STEPS: 392527.18

- tasa de descuento: 0.9

Game 1000 Mean Score 1.83 Record: 3 EXP_RATE: 0.8608649248865096 STEPS: 1851.67

Game 10000 Mean Score 17.15 Record: 16 EXP_RATE: 0.22331899945635536 STEPS: 3871.15

Game 20000 Mean Score 72.2 Record: 53 EXP_RATE: 0.049984578080854006 STEPS: 10457.22

Game 30000 Mean Score 209.62 Record: 112 EXP_RATE: 0.011308440880102967 STEPS: 25807.28

- tasa de descuento: 0.99

Game 1000 Mean Score 1.62 Record: 3 EXP_RATE: 0.8608649248865096 STEPS: 1807.41

Game 10000 Mean Score 19.36 Record: 23 EXP_RATE: 0.22331899945635536 STEPS: 4129.59

Game 20000 Mean Score 84.96 Record: 53 EXP_RATE: 0.049984578080854006 STEPS: 11907.39

Game 30000 Mean Score 193.06 Record: 111 EXP_RATE: 0.011308440880102967 STEPS: 23889.96

Vamos viendo que mientras mayor la tasa de descuento, más lento va aprendiendo. Esto tiene mucho sentido, pues como mencionamos, la tasa de descuento afecta en la recompensa a futuro, y por tanto no nos importa que aprenda lo más óptimo de la forma más rápida posible. Como en este juego nos importa simplemente que aprenda con rapidez a pasar por los arcos, tiene sentido que el mejor desempeño lo haya tenido en DR: 0.

3-. Este determina cómo va variando la actualización de un valor de Q-learning, es decir qué tan rápido convergerá a una solución y qué tanto buscará por esta. Un valor muy alto podría llevar a soluciones subóptimas, pero uno pequeño podría tomar demasiado tiempo.

LR: 0.05

Game 1000 Mean Score 1.61 Record: 3 EXP_RATE: 0.8608649248865096 STEPS: 1806.78

Game 10000 Mean Score 27.32 Record: 20 EXP_RATE: 0.22331899945635536 STEPS: 5090.84

Game 20000 Mean Score 187.45 Record: 95 EXP_RATE: 0.049984578080854006 STEPS: 23537.17

Game 30000 Mean Score 560.25 Record: 169 EXP_RATE: 0.011308440880102967 STEPS: 61183.29

LR: 0.3

Game 1000 Mean Score 2.01 Record: 4 EXP_RATE: 0.8608649248865096 STEPS: 1868.1

Game 10000 Mean Score 28.8 Record: 20 EXP_RATE: 0.22331899945635536 STEPS: 5266.78

Game 20000 Mean Score 183.57 Record: 131 EXP_RATE: 0.049984578080854006 STEPS: 23113.63

Game 30000 Mean Score 577.65 Record: 179 EXP_RATE: 0.011308440880102967 STEPS: 62783.41

LR: 0.9

Game 1000 Mean Score 1.85 Record: 4 EXP_RATE: 0.8608649248865096 STEPS: 1851.03

Game 10000 Mean Score 30.62 Record: 29 EXP_RATE: 0.22331899945635536 STEPS: 5488.66

Game 20000 Mean Score 184.48 Record: 109 EXP_RATE: 0.049984578080854006 STEPS: 23220.33

Game 30000 Mean Score 570.99 Record: 180 EXP_RATE: 0.011308440880102967 STEPS: 62127.15

Vemos que aumentar el learning rate mejora bastante la velocidad con que el modelo encuentra la solución. Esto, de nuevo, era esperable, ya que como dijimos, este parámetro definía la velocidad de convergencia. Sin embargo, no hubo una gran diferencia entre LR 0.3 y LR 0.9, por lo que quizá el modelo también comenzó a tomar decisiones subóptimas en este gap. Por otro lado, el tiempo de computo sí fue menor en el último caso, lo cual es favorable, ya que se obtuvieron resultados bastante similares en un tiempo menor.

Actividad 4:

Se me ocurre que se podría recompensar al agente con las mismas ponderaciones, pero restando 50 puntos por partida perdida, de modo que el agente vea la pérdida del juego como algo que realmente no puede ocurrir, ya que esto es algo mucho más de peso que retroceder (lo cual también resta un punto) en cuestión de lo que queremos que el código haga. Aumentaría un poco más también la ponderación de pasar por un arco, pues en cierto punto, el score sigue mejorando, pero el record de arcos casi ya no, dando a entender que muchos puntos se ganan por ir optimizando el camino e ir retrocediendo menos, pero no por pasar por estos arcos. También aumentaría la penalización del juego a retrocer a 5 puntos, pues no queremos para nada que este retroceda (ya que así no es como funciona el juego realmente).